IN THE CLAIMS

Please amend the claims as follows:

1. - 4. (Canceled)

5. (Currently Amended) The method of claim [[4]] 6, wherein the Digital Signal Processor resources are selected from the group consisting of registers, pipeline structures, instruction scheduling, memory, and MAC units.

6. (Currently amended) A method, comprising:

modifying source code, including multiple instructions, by performing a Lexical Functional Grammar Analysis (LFGA) operation on one or more instructions as a function of a Digital Signal Processor (DSP) architecture wherein modifying the source code by performing Lexical Functional Grammar Analysis operation further comprises:

analyzing the modified source code for syntax and semantic; and

further modifying the source code based on the outcome of the analysis; and

generating assembly code using the modified source code ~~The method of claim 4,~~ wherein selecting and comparing one or more instructions in the intermediate code using the Genetic algorithm comprises:

selecting an instruction in the intermediate code to generate the efficient code;

computing a current number of clock cycles required to execute the selected instruction;

computing reduction in number of clock cycles by using the current and previous computed number of clock cycles;

comparing the reduction in the number of clock cycles with a predetermined reduction in number of clock cycles;

if the reduction in the number of clock cycles is greater than the predetermined reduction in the number of clock cycles, then selecting a relevant instruction based on specific Digital Signal Processor architecture and applying cross over for the selected instruction and repeating the computing number of clock cycles, the computing reduction

in number of clock cycles, and the comparing operations until the reduction in the number of clock cycles is less than or equal to the predetermined reduction in the number of clock cycles; ~~and~~

if the reduction in the number of clock cycles is less than the predetermined reduction in the number lock cycles, then selecting the instruction for the efficient code and repeating the above operations for a next instruction in the intermediate code: and

changing structure of the modified source code through a series of iterations using Finite State Morphology (FSM) to generate efficient source code wherein modifying the structure of the modified source code using Finite State Morphology comprises:

generating intermediate code by rearranging concurrent distributed instructions based on Digital Signal Processor resources using Petri Nets algorithm; and

generating the efficient code by selecting and comparing one or more instructions in the intermediate code to one or more other similar available instructions using Genetic algorithm

7. – 14. (Canceled)

15. (Currently Amended) A method of generating assembly code for execution by a DSP, comprising:

receiving source code in higher level language including multiple instructions;

parsing the source code using Lexical Functional Grammar Analysis to modify one or more instructions in the program such that the modified instructions comply with specific Digital Signal Processor resources wherein the specific Digital Signal Processor resources are selected from the group consisting of registers, pipeline structures, instruction scheduling, memory, ALUs, and MAC units.

analyzing the parsed source code for syntax;

updating the parsed source code for syntax based on the analysis;

generating intermediate code by rearranging concurrent distributed instructions based on the specific Digital Signal Processor resources using Petri Nets algorithm;

generating first efficient code by selecting and comparing one or more instructions in the intermediate code to one or more other similar instructions, available to process on the specific DSP, using Genetic algorithm ~~The method of claim 13,~~ wherein selecting and comparing one or more instructions in the intermediate code using the Genetic algorithm comprises:

selecting an instruction in the intermediate code to generate the efficient code;

computing a current number of clock cycles required to execute the selected instruction;

computing reduction in number of clock cycles by using the current and previous computed number of clock cycles;

comparing the reduction in number of clock cycles with a predetermined reduction in number of clock cycles;

if the reduction in the number of clock cycles is greater than the predetermined reduction in the number of clock cycles, then selecting another instruction similar to the selected instruction based on the specific Digital Signal Processor architecture and applying cross over for the selected instruction and repeating the computing number of clock cycles, the computing reduction in number of clock cycles, and the comparing operations until the reduction in the number of clock cycles is less than or equal to the predetermined reduction in the number of clock cycles; and

if the reduction in the number of clock cycles is less than the predetermined reduction in the number of clock cycles, then selecting the instruction for the efficient code and repeating the above operations for a next instruction in the intermediate code;

selecting one or more instructions from the multiple instructions that have similar available instruction sets in the first efficient code;

generating second efficient code by performing dynamic instruction replacement on the one or more selected instructions; and

generating the assembly code by mapping the second efficient code to assembly language code.

16. – 30. (Canceled)